3. (Amended) [A] The method as claimed in [any one of] claim[s] 1 [or 2] wherein the state of the software object [may] corresponds to the current values held in [the]a stack, a heap, and global variables [, registers, program counter and the like,]of the software object.

4. (Amended) [A] The method [as claimed in any preceding] of claim 1 or 2 or 3 wherein the watermark is stored in an [object's] execution state of the software object whereby [an]the input sequence [*I*] is constructed which, when fed to an application of which the software object is a part, will make the software object [*O*] enter a second state which [represents]is a representation of the watermark, the representation being validated or checked by examining the stack, heap, global variables, registers, program counter and the like, of the software object [*O*].

5. (Amended) [A] The method [as claimed in any one] of claim[s] 1 or 2 wherein the watermark is embedded in [the] an execution trace of the software object [*O*] whereby, as a special input [*I*] is fed to [*O*]the software object, [the] an address/operator trace is monitored and, based on a property of the trace, [a]the watermark is extracted.

6. (Amended) [A] The method [as claimed in any one] of claim[s] 1 [to 4] or 2 or 3 wherein the watermark is embedded in [the] a topology of a dynamically built graph structure.

7. (Amended) [A] The method [as claimed in] of claim 6 wherein the dynamically built graph structure [(or watermark graph)] corresponds to a representation of [the]a data structure of the program and may be viewed as a set of nodes together with a set of vertices.

8. (Amended) [A] The method [as claimed in any preceding] of claim 1, or 2 or 3 further comprising the step of:

(c) building a recognizer [*R*] concurrently with the input [*I*] sequence and the watermark [*W*].

9. (Amended) [A] The method [as claimed in] of claim 8 wherein [*R*] the recognizer is a function adapted to identify and extract the watermark [graph] from all other dynamic structures on [the] a heap or stack.

10. (Amended) [A] The method [as claimed in either] of claim 8 [or 9] wherein the watermark [*W*] incorporates a marker that will allow [*R*] the recognizer to recognize it easily.

11. (Amended) [A] The method [as claimed in any one] of claim[s] 8 [to 10] wherein [*R*] the recognizer is retained separately from the program and whereby [*R*] the recognizer inspects the state of the program.

12. (Amended) [A] The method [as claimed in any one] of claim[s] 8 [to 11] wherein [*R*] the recognizer is dynamically linked with the program when it is checked for the existence of a watermark.

13. (Amended) [A] The method [as claimed in any preceding] of claim 1, or 2, or 3 wherein [the application of which] the software object [forms a] is part of an application that is obfuscated or incorporates tamper-proofing code.

14. (Amended) [A] The method [as claimed in any one] of claim[s] 8 [to 12] wherein [*R*] the recognizer checks [*W*] the watermark for a signature property [*s(W)*].

15. (Amended) [A] The method [as claimed in] of claim 14 wherein the signature property [s(*W*)] is evaluated by testing for a specific result from a hard computational problem.

16. (Amended) [A] The method [as claimed in either] of claim 14 [or claim 15] including the step of:

(d) [the creation of] creating a number having at least one numeric property[*n*] which [may be] is embedded in the topology of [*W*] the watermark, whereby the signature property [may be] is evaluated by testing [one or more] the at least one numeric property[properties of *n*].

17. (Amended) [A] The method [as claimed in] of claim 16 wherein the signature property is evaluated by testing whether [*n*] the number is [the] a product of two primes.

18. (Amended) A method of verifying the integrity or origin of a program including the steps of:

(a) watermarking the program with a watermark [*W*], wherein the watermark [*W*] is stored in the state of a program as the program is being run with an [particular] input sequence[*I*];

(b) building a recognizer [*R*] concurrently with the input [*I*] sequence and watermark *W* wherein the recognizer is adapted to extract the watermark [graph] from other dynamically allocated data wherein [*R*] the recognizer is kept separately from the program; wherein [*R*]the

recognizer is adapted to check for a number[*n*].

19. (Amended) [A] The method [of verifying the integrity of origin of a program as claimed in] of claim 18, wherein the number [*n*] is the product of two primes and wherein [*n*] the number is embedded in [the] a topology of [*W*] the watermark.

20. (Amended) [A] The method [as claimed in either] of claim 18 or claim 19 wherein the number [*n*] is derived from a[ny] combination of numbers depending on [the] a context and an application for the watermark.

21. (Amended) [A] The method [as claimed in any one] of claim[s] 18 or 19 [to 20] wherein the program [or code] is further adapted to be resistant to tampering, the resistance to tampering capable of being [preferably]by means of obfuscation or by adding tamper-proofing code.

22. (Amended) [A] The method [as claimed in any one] of claim[s] 18 or 19 [to 21] wherein the recognizer [*R*] checks for the effect of the watermark[ing code] on [the] an execution state of the [application]program, thereby preserving [the] an ability to recognize the watermark [in cases] where semantics-preserving transformations have been applied to the [application]program.

23. (Amended) A method of watermarking software including the steps of:

(a) embedding a watermark in a static string; and

(b) applying an obfuscation technique whereby [this]the static string is converted into executable code.

24. (Amended) A method of watermarking software comprising the steps of:

(a) [wherein the]choosing a watermark [*W* is chosen] from a class of graphs [*G*]having a plurality of members, [wherein] each member of [*G*] the class of graphs has [one or more properties, such as planarity] at least one property, [said] the at least one property being capable of being tested by integrity-testing software; and

(b) applying the watermark to the software.

25. (Amended) [A] The method of [watermarking software as claimed in] claim 24 wherein the watermark is rendered tamperproof to certain transformations by subjecting the watermark graph to one or more local transformations.

26. (Amended) [A] The method of [watermarking software as claimed in] claim 25 wherein the watermark is a watermark graph including at least one node and wherein each of the at least one node of the watermark graph is expanded into a cycle.

27. (Amended) A method of fingerprinting software comprising the steps of:

(a) providing [wherein] a plurality of watermarked programs, [obtained as claimed in any preceding claim are produced] the plurality of watermarked programs being obtained by providing an input sequence for each program of the plurality of programs and storing a watermark in a state of a software object for the program as the software object is being run with

the input sequence.

28. (Amended) [A]The method of fingerprinting software as claimed in claim 27 wherein the plurality of watermarked programs each of which has a number [n] with a common prime factor [p].

Please cancel claim 29.

30. (Amended) [Software written to perform the method as claimed in any preceding claim]A computer readable medium including a program for watermarking a software object, the program including instructions for:

(a) providing an input sequence; and

(b) storing a watermark in the state of the software object as the software object is being run with the input sequence.

31. (Amended) A computer [programmed to perform the method as claimed in any one of claims 1 to 27] comprising:

a software object;

an input sequence; and

a watermark stored in the state of the software object as the software object is being run with the input sequence.

Please add the following claims:

32. A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, the plurality of watermarked programs being obtained by watermarking each program of the plurality of programs with a watermark, wherein the watermark is stored in the state of a program as the program is being run with an input sequence and building a recognizer concurrently with the input sequence and watermark W wherein the recognizer is adapted to extract the watermark from other dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number.

33. A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, the plurality of watermarked programs being obtained by watermarking each program of the plurality of programs with a watermark, the watermark being obtained by embedding a watermark in a static string and applying an obfuscation technique whereby the static string is converted into executable code.

34. A method of fingerprinting software comprising the steps of:

(a) providing a plurality of watermarked programs, the plurality of watermarked programs being obtained by watermarking each program of the plurality of programs with a watermark, the watermark being obtained by choosing a watermark from a class of graphs having a plurality of members, each member of the class of graphs has at least one property, the at least one property being capable of being tested by integrity-testing software and applying the watermark to the software.

35. A computer-readable medium including a program for verifying the integrity or origin of a program, the program including instructions for:

(a) watermarking the program with a watermark, wherein the watermark is stored in the state of a program as the program is being run with an input sequence;

(b) building a recognizer concurrently with the input sequence and watermark wherein the recognizer is adapted to extract the watermark from other dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number.

36. A computer-readable medium including a program for watermarking software, the program including instructions for:

(a) embedding a watermark in a static string; and

(b) applying an obfuscation technique whereby the static string is converted into executable code.

37. A computer-readable medium including a program for watermarking software, the program including instructions for:

(a) choosing a watermark from a class of graphs having a plurality of members, each member of the class of graphs has at least one property, the at least one property being capable of being tested by integrity-testing software; and

(b) applying the watermark to the software.

38. A computer capable of verifying the integrity or origin of a program, the computer

comprising:

an input sequence;

a watermark for watermarking the program, wherein the watermark is stored in the state of a program as the program is being run with the input sequence;

a recognizer built concurrently with the input sequence and watermark wherein the recognizer is adapted to extract the watermark from other dynamically allocated data wherein the recognizer is kept separately from the program; wherein the recognizer is adapted to check for a number.

39. A computer for watermarking software comprising:

a static string;

a watermark embedded in the static string; and

an obfuscation technique for converting the static string into executable code.

40. A computer comprising:

a watermark from a class of graphs having a plurality of members, each member of the class of graphs has at least one property, the at least one property being capable of being tested by integrity-testing software; and

software to which the watermark is applied.